

**AMENDMENTS TO THE CLAIMS**

1. (Original) A method, comprising:  
storing one or more tasks in a queue, wherein each task has an associated exit routine;  
determining at least one task to process based on a priority scheme;  
processing the at least one task; and  
calling the exit routine based on determining that the task has not completed processing within a  
preselected period of time.
2. (Original) The method of claim 1, wherein storing the one or more task in the queue  
comprises storing at least one task in the queue at every preselected time interval.
3. (Original) The method of claim 1, wherein storing the one or more tasks in the queue  
comprises generating an interrupt and storing the one or more tasks in the queue in response to  
detecting the interrupt.
4. (Original) The method of claim 1, wherein determining at least one task to process based  
on the priority scheme comprises determining the at least one task based on a first-in, first-out  
priority scheme.
5. (Original) The method of claim 1, wherein calling the exit routine comprises calling the  
exit routine if the task does not complete executing within a preselected amount of time.

6. (Original) The method of claim 1, wherein calling the exit routine comprises terminating the task currently processing and returning control to a task picker in the queue.
7. (Original) The method of claim 1, wherein processing the at least one task comprises executing the task and programming a timer to generate an interrupt after a preselected time, wherein the preselected time substantially corresponds to the amount of time required for the task to complete executing.
8. (Original) An apparatus, comprising:  
a queue having a task picker stored therein; and  
a controller communicatively coupled to the queue, the controller to:  
determine if at least one task other than the task picker is stored in the queue;  
execute the task based on determining that at least one task other than the task picker is stored in the queue; and  
execute the task picker in response to executing the task and continue executing the task picker until a preselected event occurs.
9. (Original) The apparatus of claim 8, wherein the preselected event comprises detection of an interrupt.
10. (Original) The apparatus of claim 8, wherein the preselected event comprises detection of another task being present in the queue.

11. (Original) The apparatus of claim 8, wherein each task stored in the queue comprises an exit routine to terminate that task.

12. (Original) The apparatus of claim 8, wherein the controller comprises a controller of a south bridge in a computer system.

13. (Original) The apparatus of claim 8, wherein the controller determines that more than one task is stored in the queue and wherein the controller selects a task to execute from the one or more tasks based on a priority scheme.

14. (Original) The apparatus of claim 13, wherein the priority scheme is a first-in, first-out scheme.

15. (Original) An article comprising one or more machine-readable storage media containing instructions that when executed enable a processor to:

store one or more tasks in a storage space, wherein each task has an associated exit routine;

determine at least one task to process based on a priority scheme;

process the at least one task; and

call the exit routine based on determining that the task cannot be processed to completion.

16. (Original) The article of claim 15, wherein the instructions when executed enable the processor to store the one or more tasks in the storage space comprises storing at least one task in the storage space at every preselected time interval.
17. (Original) The article of claim 15, wherein the instructions when executed enable the processor to generate an interrupt and store the one or more tasks in the storage space in response to detecting the interrupt.
18. (Original) The article of claim 15, wherein the instructions when executed enable the processor to determine the at least one task based on a first-in, first-out priority scheme.
19. (Original) The article of claim 15, wherein the instructions when executed enable the processor to call the exit routine if the task does not complete executing within a preselected amount of time.
20. (Original) The article of claim 15, wherein the instructions when executed enable the processor to terminate the task currently processing and return control to a task picker in the storage space.
21. (Original) The article of claim 15, wherein the instructions when executed enable the processor to execute the task and to program a time to generate an interrupt at a preselected time, wherein the preselected time is greater than the time required for the task to complete executing.

22. (Original) An apparatus, comprising:
- a queue having a task picker stored therein, the queue adapted to store one or more tasks;
  - a failure recovery timer to generate an interrupt at preselected time intervals, wherein each preselected time interval is greater than the time it takes for each of the tasks stored in the queue to execute; and
  - a controller adapted to:
    - select a task to execute based on a priority scheme
    - execute the task;
    - determine if the task completes execution within the preselected time interval;
    - terminate the task in response to determining that the task failed to complete within the preselected time interval; and
    - execute the task picker in response to terminating the task.
23. (Original) The apparatus of claim 22, wherein the priority scheme is based on a first-in, first-out scheme;
24. (Original) The apparatus of claim 22, wherein each task has an associated exit routine and wherein the controller terminates the task by calling the exit routine.
25. (Original) The apparatus of claim 22, wherein the controller resets the failure recovery timer before executing the task.

26. (Original) The apparatus of claim 22, wherein the controller determines if the task completes execution within the preselected time interval comprises:
- detecting a first failure recovery interrupt;
  - causing an interrupt service routine to determine a task ID associated with a task executing at the time of the first failure recovery interrupt;
  - logging the determined task ID;
  - detecting a second failure recovery interrupt;
  - determining a task ID associated with a task executing at the time of the second failure recovery interrupt; and
  - terminating the task executing at the time of the second failure recovery interrupt in response to determining that the two task IDs are the same.
27. (Original) The apparatus of claim 22, further comprising a repetitive timer for generating interrupts on a periodic basis, wherein the controller posts a task in the queue in response to detecting an interrupt generated by the repetitive timer.
28. (Original) The apparatus of claim 22, wherein the controller resets the failure recovery timer before executing the task picker.
29. (Canceled)